

▼ Unit 02, 03 실습

구글 드라이브 연동 & CSV 파일 데이터 읽어오기

```
1 #구글 드라이브와 연동
2 from google.colab import drive
3 drive.mount('/content/drive')
4 directory_path = '/content/drive/MyDrive/2021-1학기/파데분/실습/'
5
6 # 자신의 구글 드라이브 directory_path 위치에 seoul.csv 파일 저장
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c

csv.reader() : CSV 파일에서 데이터를 읽어오는 함수

csv.writer() : CSV 파일에 데이터를 저장하는 함수

헤더(header): 데이터의 속성을 설명, 첫번째 줄에 위치

next(): 첫 번째 행부터 데이터를 읽어오며 데이터의 탐색 위치를 다음 행으로 이동시키는 함수

```
1 import csv
2 #seoul.csv 데이터 한 행(row)씩 읽어오기
3 f = open(directory_path + 'seoul.csv', 'r', encoding='cp949')
4 data = csv.reader(f, delimiter=',')
5 #next() 함수를 활용해 헤더 저장하기
6 header = next(data)
7 for row in data:
8     print(row)
```

```
['2017-10-13', '108', '12.8', '6.1', '18.9']
['2017-10-14', '108', '14.4', '9', '20.5']
['2017-10-15', '108', '15.8', '9', '23']
['2017-10-16', '108', '16.6', '13.6', '22']
['2017-10-17', '108', '16.2', '9.2', '23.9']
['2017-10-18', '108', '16.5', '14.2', '19.1']
['2017-10-19', '108', '17', '11.9', '23.2']
['2017-10-20', '108', '17', '11.1', '24.2']
['2017-10-21', '108', '17.6', '11', '25.2']
['2017-10-22', '108', '16.3', '11', '23.5']
['2017-10-23', '108', '13.7', '8.6', '20.6']
['2017-10-24', '108', '14.4', '9.9', '20.9']
['2017-10-25', '108', '14.4', '9.5', '20.6']
['2017-10-26', '108', '14.5', '10.7', '18.1']
['2017-10-27', '108', '16.9', '11.9', '24.2']
['2017-10-28', '108', '17.2', '10.6', '24.8']
['2017-10-29', '108', '11.5', '6.3', '17.2']
```

```
[ '2017-10-29', '108', '11.0', '0.0', '17.2' ]
[ '2017-10-30', '108', '7.7', '2.5', '13.2' ]
[ '2017-10-31', '108', '10.4', '3.6', '16' ]
[ '2017-11-01', '108', '14.4', '11.5', '18.1' ]
[ '2017-11-02', '108', '15.5', '14', '18' ]
[ '2017-11-03', '108', '12.3', '6.1', '16.6' ]
[ '2017-11-04', '108', '7.8', '4.5', '13.7' ]
[ '2017-11-05', '108', '8.4', '2.6', '15.4' ]
[ '2017-11-06', '108', '11.4', '8', '16.5' ]
[ '2017-11-07', '108', '14.4', '9.2', '18.4' ]
[ '2017-11-08', '108', '13.1', '6.9', '17.2' ]
[ '2017-11-09', '108', '9.4', '3.7', '16.8' ]
[ '2017-11-10', '108', '9.8', '5.3', '15.9' ]
[ '2017-11-11', '108', '5.9', '2.7', '10.7' ]
[ '2017-11-12', '108', '5.3', '0.3', '11.4' ]
[ '2017-11-13', '108', '8.4', '3.1', '11.8' ]
[ '2017-11-14', '108', '7.5', '2.9', '11.4' ]
[ '2017-11-15', '108', '3.2', '-1.1', '6.5' ]
[ '2017-11-16', '108', '1', '-3.4', '6.4' ]
[ '2017-11-17', '108', '3', '-1.6', '5.8' ]
[ '2017-11-18', '108', '-2.1', '-4.8', '2.6' ]
[ '2017-11-19', '108', '-1.6', '-6.6', '3.5' ]
[ '2017-11-20', '108', '-0.4', '-3.9', '4.2' ]
[ '2017-11-21', '108', '1.3', '-4.2', '8.1' ]
[ '2017-11-22', '108', '4.3', '0.6', '10.8' ]
[ '2017-11-23', '108', '0.9', '-2.5', '4.7' ]
[ '2017-11-24', '108', '-0.4', '-2.5', '2.9' ]
[ '2017-11-25', '108', '3', '-3.3', '7.3' ]
[ '2017-11-26', '108', '3.3', '-0.3', '7.6' ]
[ '2017-11-27', '108', '2.5', '-2.9', '9.5' ]
[ '2017-11-28', '108', '6.4', '1.3', '11.2' ]
[ '2017-11-29', '108', '3.2', '-1.8', '8.1' ]
[ '2017-11-30', '108', '-2.4', '-5.2', '2.7' ]
[ '2017-12-01', '108', '-2.2', '-7.6', '3.1' ]
[ '2017-12-02', '108', '1.9', '-4.6', '8.1' ]
[ '2017-12-03', '108', '4.9', '3.1', '8.1' ]
[ '2017-12-04', '108', '-1.2', '-6.1', '4.5' ]
[ '2017-12-05', '108', '-4.2', '-8.2', '-0.4' ]
[ '2017-12-06', '108', '0.2', '-4.5', '6' ]
[ '2017-12-07', '108', '0.7', '-3.5', '4.6' ]
[ '2017-12-08', '108', '-3.8', '-7.1', '-0.5' ]
[ '2017-12-09', '108', '-0.5', '-5.6', '5.4' ]
[ '2017-12-10', '108', '1.1', '-2.5', '6.7' ]
[ '2017-12-11', '108', '-7', '-11', '-2.5' ]
```

▼ 실습1:

질문 : 기상 관측 이래, 서울의 최고 기온이 가장 높았던 날은 언제였고, 몇 도였을까?

질문을 해결하는데 필요한 절차(알고리즘)는?

1 | 데이터를 읽어오다.

```
1 #step 1) 데이터 불러서 한 행씩 출력하기
2 #step 2) 데이터 중 최고 기온을 실수로 변환하여 한 행씩 출력하기
3
4 f = open(directory_path + 'seoul.csv', 'r', encoding='cp949')
5 data = csv.reader(f)
6
7 header = next(data)
8 for row in data:
9     if row[-1] == '':
10         row[-1] = -999
11     row[-1] = float(row[-1])
12     print(row[-1])
13
14
15 f.close()
```

```
20.2
19.3
20.3
20.6
22.8
25.3

29.8
25.6
24.7
26.5
27.7
26.7
13.3
19.7
15.6
17.6
19.3
16.1
19.6
23.3
20.7
25.7
25.3
23.6
19.6
23.0
24.5
22.8
18.5
20.1
23.5
24.0
25.0
27.7
28.2
30.2
26.7
26.6
24.2
22.2
22.6
```

23.6
29.1
27.1
25.1
27.7
28.0
23.1
26.8
27.5
31.3
31.8
24.3
25.2
26.3
27.2
29.3
29.5
29.1

```
1 #step 3) 최고 기온과 최고 기온이었던 날짜 찾기
2
3 max_temp = -999 # 최고 기온 값을 저장할 변수
4 max_date = '' # 최고 기온이 가장 높았던 날짜를 저장할 변수
5 f = open(directory_path + 'seoul.csv', 'r', encoding='cp949')
6 data = csv.reader(f)
7 header = next(data)
8
9 for row in data:
10     if row[-1] == '':
11         row[-1] = -999
12     row[-1] = float(row[-1])
13     if max_temp < row[-1]:
14         max_date = row[0]
15         max_temp = row[-1]
16
17 f.close()
18 print('기상 관측 이래 서울의 최고 기온이 가장 높았던 날은', max_date + '로,', max_temp, '도 였습니C
19
```

기상 관측 이래 서울의 최고 기온이 가장 높았던 날은 1994-07-24로, 38.4 도 였습니다.

▼ Unit 04, 05 실습

matplotlib: 2D 형태의 그래프, 이미지 등을 그릴 때 사용

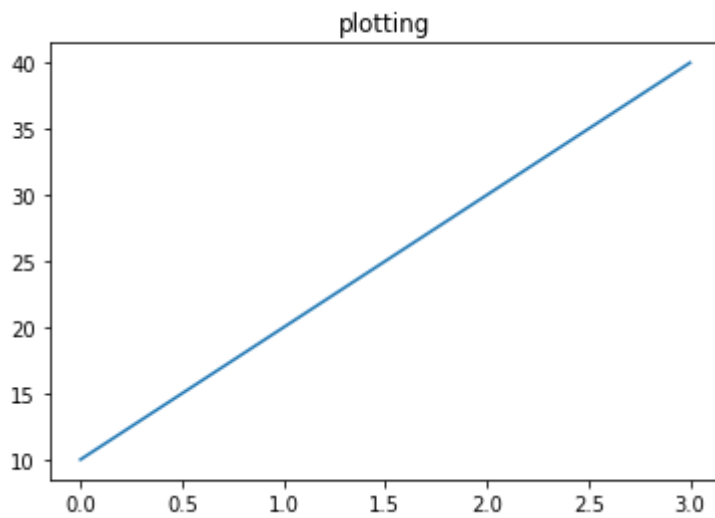
```
1 #matplotlib 라이브러리의 pyplot 모듈을 'plt' 라는 별명으로 부르기(as, alias)
2 import matplotlib.pyplot as plt
3
```

▼ 실습 2: 그래프에 옵션 추가하기

```

1 #그래프에 제목 넣기
2
3 plt.title('plotting')
4 plt.plot([10, 20, 30, 40])
5 plt.show()

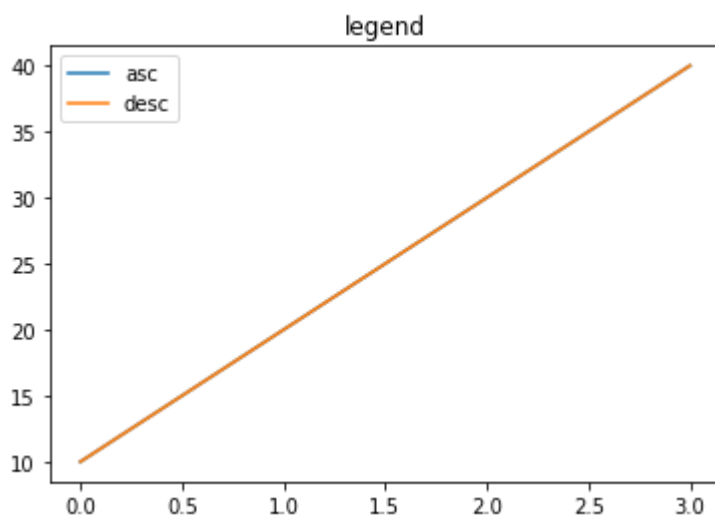
```



```

1 #그래프에 범례 넣기
2 plt.title('legend')
3 plt.plot([10,20,30,40], label = 'asc') #증가
4 plt.plot([10,20,30,40], label = 'desc') #감소
5 plt.legend()
6 plt.show()

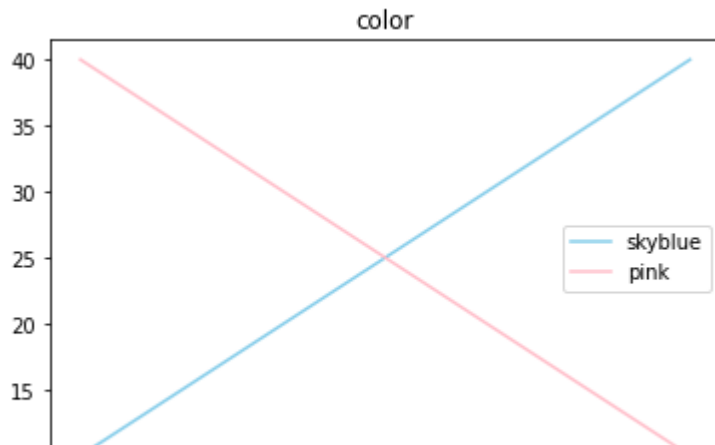
```



```

1 #그래프 색상 바꾸기
2 plt.title('color') # 제목 설정
3 plt.plot([10,20,30,40], color = 'skyblue', label='skyblue') # 하늘색 그래프
4 plt.plot([40,30,20,10], color = 'pink', label='pink') # 분홍색 그래프
5 plt.legend() # 범례 표시
6 plt.show()

```



1 #그래프 선 모양 바꾸기

2

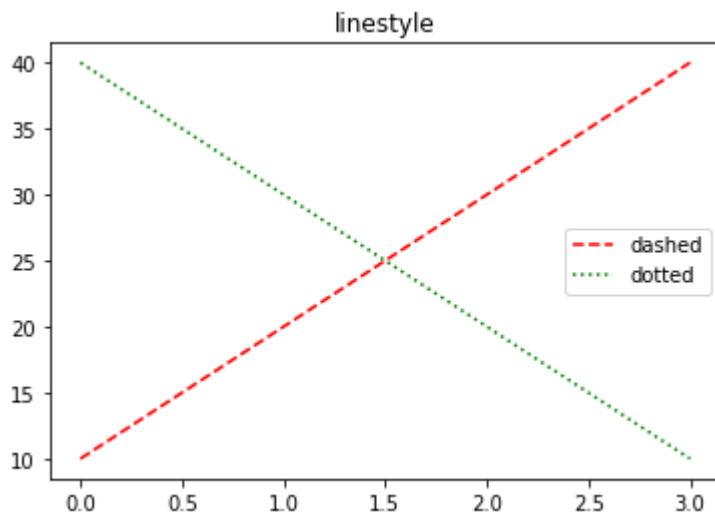
3 plt.title('linestyle') #제목 설정

4 plt.plot([10,20,30,40], color='r', linestyle='--', label='dashed') # 빨간색 dashed 그래프

5 plt.plot([40,30,20,10], color='g', ls=':', label='dotted') # 초록색 dotted 그래프

6 plt.legend() # 범례 표시

7 plt.show()



1 # 그래프 마커 모양 바꾸기

2 plt.title('marker') # 제목 설정

3 plt.plot([10,20,30,40], 'r.', label='circle') # 빨간색 원형 마커 그래프





4 plt.plot([40,30,20,10], 'g^', label='triangle up') # 초록색 삼각형 마커 그래프

5 plt.legend() # 범례 표시

6 plt.show()



▼ 실습3: 날짜 데이터 추출하기

```
25 |     |
1
2 date = '1907-10-01'
3 print(date.split('-'))
4
5 print(date.split('-')[0])
6 print(date.split('-')[1])
7 print(date.split('-')[2])
8
9

['1907', '10', '01']
1907
10
01
```

▼ 실습4: 내 생일의 최고 기온 및 최저 기온 데이터 시각화하기

본인의 생일을 사용하여 구현

생일: 10월 08일

구글 코랩 한글 사용

1. 폰트 설치
2. 런타임 - 런타임 다시 시작 을 클릭하여 런타임을 재시작

```
1 # 폰트 설치
2 !sudo apt-get install -y fonts-nanum
3 !sudo fc-cache -fv
4 !rm ~/.cache/matplotlib -rf
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 9,604 kB of archives.
After this operation, 29.5 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-nanum all 20170925-1 [9,604
```

```

Fetched 9,604 kB in 1s (9,691 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-nanum.
(Reading database ... 160690 files and directories currently installed.)
Preparing to unpack ../fonts-nanum_20170925-1_all.deb ...
Unpacking fonts-nanum (20170925-1) ...
Setting up fonts-nanum (20170925-1) ...
Processing triggers for fontconfig (2.12.6-0ubuntu2) ...
/usr/share/fonts: caching, new cache contents: 0 fonts, 1 dirs
/usr/share/fonts/truetype: caching, new cache contents: 0 fonts, 3 dirs
/usr/share/fonts/truetype/humor-sans: caching, new cache contents: 1 fonts, 0 dirs
/usr/share/fonts/truetype/liberation: caching, new cache contents: 16 fonts, 0 dirs
/usr/share/fonts/truetype/nanum: caching, new cache contents: 10 fonts, 0 dirs
/usr/local/share/fonts: caching, new cache contents: 0 fonts, 0 dirs
/root/.local/share/fonts: skipping, no such directory
/root/.fonts: skipping, no such directory
/var/cache/fontconfig: cleaning cache directory
/root/.cache/fontconfig: not cleaning non-existent cache directory
/root/.fontconfig: not cleaning non-existent cache directory
fc-cache: succeeded

```

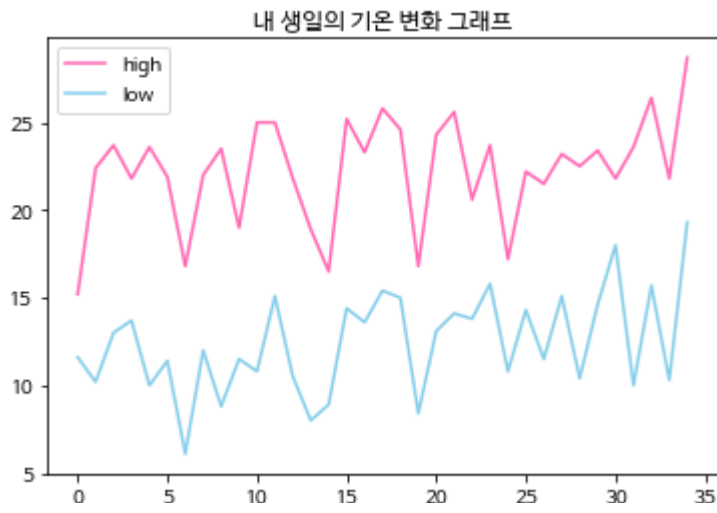
```

1 import csv
2 import matplotlib.pyplot as plt
3
4 f = open(directory_path + 'seoul.csv', 'r', encoding='cp949')
5 data = csv.reader(f)
6 next(data)
7 high = [] # 최고 기온 값을 저장할 리스트 high 생성
8 low = [] # 최저 기온 값을 저장할 리스트 low 생성
9
10 '''
11 code 추가
12 '''
13 for row in data:
14     if row[-1] != '' and row[-2] != '':
15         date = row[0].split('-')
16         if 1983 <= int(date[0]):
17             if date[1] == '10' and date[2] == '08':
18                 high.append(float(row[-1]))
19                 low.append(float(row[-2]))
20
21 plt.rc('font', family='NanumBarunGothic')
22 plt.rcParams['axes.unicode_minus'] = False
23 plt.title('내 생일의 기온 변화 그래프')
24 plt.plot(high, 'hotpink', label = 'high')
25 plt.plot(low, 'skyblue', label = 'low')
26 plt.legend() # 범례 표시
27 plt.show() # 그래프 나타내기

```

최고 기온 값과 최저 기온 값이 존재한다면
날짜 값을 - 문자를 기준으로 구분하여 저장
1983년 이후 데이터라면
10월 08일이라면
최고 기온 값을 high 리스트에 저장
최저 기온 값을 low 리스트에 저장

나눔 고딕을 기본 글꼴로 설정
마이너스 기호 깨짐 방지
제목 설정
high 리스트에 저장된 값을 hotpink 색으로 그리
low 리스트에 저장된 값을 skyblue 색으로 그리



===== 실습 끝